Ronak Patel
GT ID: rpatel88
CS7641: Assignment 4

Solving Markov Decision Processes using Value Iteration, Policy Iteration, and Q-Learning

Problem Overview

A Markov Decision Process (MDP) is a decision-making framework used in stochastic environments to find a decision-making policy. A policy includes a representation of state-action pairs for every possible state in an environment. An MDP typically involves a reward system, where an agent, which is navigating through the environment, is supplied with a reward for each action taken through each state. The goal when solving an MDP is to maximize the rewards taken for each action. The optimal policy will provide a schema for the best actions an agent can take in each state to maximize rewards. In this analysis, two different MDP problems will be evaluated and solved using MDP and reinforcement learning methods. The two MDP problems being evaluated are the cliff walking problem and the frozen lake problem, both are described in greater detail below.

4x12 Cliff Walking Problem

In the 4x12 cliff walking problem illustrated in Figure 1, an agent must ultimately try reaching the goal state, denoted by the green square, as efficiently as possible while avoiding the cliff. The agent is allowed four possible actions: move up, down, left, or right. The agent starts at a start space (black square) and receives a step reward throughout the navigation process unless it reaches the goal state or the cliff state (red squares), both of which are terminating states which end the game. To incentivize the agent, a small negative reward is taken for each step taken; whereas the cliff and goal states are large negative and positive reward respectively. Furthermore, there is a small set probability that "wind" may influence an agent's movement and blow the agent down one space regardless of the desired action taken by the learner.

This problem is interesting because the answer to the problem is intuitive and allows easy analysis to see if the learner has converged to the optimal policy. Furthermore, we can see how the stochastic nature of the problem effects the leaners ability to solve the problem in an infinite horizon. If the wind probability is set high enough, we can assume that the agent must move up to the top of the grid first and then across and down to the goal. A lower wind probability will allow the agent to walk on the path directly above the cliff and into the goal. This problem can also represent an elementary self-guided robotic navigation system in the real-world, where the robot must navigate an environment safely by overcoming obstacles and unaccounted environmental factors.

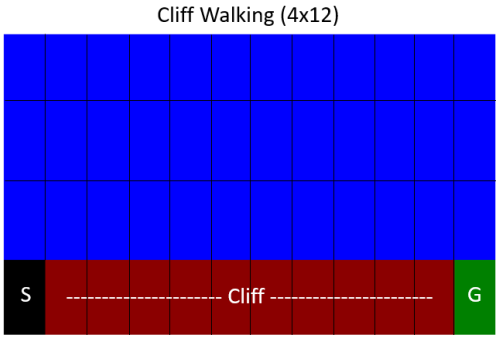


Figure 1. Cliff Walking Environment

Frozen Lake Problem

The Frozen Lake problem, illustrated in Figures 2 and 3, consists of an agent whose goal is to find the optimal path which maximizes reward by reaching the goal state. The agent has four possible actions: move up, down, left, or right and is navigating through a grid environment which is deemed "slippery", meaning that the agent may be forced to deviate from the action it would like to take. In the case that the agent slips from it's wanted direction, the agent will move perpendicular to its desired direction with equal probability on each side. The environment also has "holes" throughout the grid (denoted by red squares). A hole is a terminating state for the game and penalizes the agent heavily. The agent starts at the start space, denoted by the black square, and must move to the goal state, denoted by a green square. The agent is incentivized to move towards the goal space by collecting a small negative reward for each step it takes; similarly, the agent collects a large positive reward for reaching the goal state. For this experiment, 8x8 and 15x15 environment sizes will be tested.

The frozen lake problem is interesting because it can represent primitive robotic navigation algorithms as it allows and forces for the agent to explore multiple paths and directions to reach the goal state. This is attributed to the holes which are dispersed throughout the environment which can be viewed as obstacles, and they force the agent to navigate around the holes. The problem also introduces stochasticity in two directions which represent unforeseen environmental impacts on the agent, and it will be interesting to see how the agent recognizes and deals with unexpected actions and outcomes. Finally, the hypothesis space and solution space for the frozen lake problem is much bigger than the cliff walking problem, allowing us to catch sub-optimal policies, their mishaps, and the learners' understanding of the environment given a specified set of hyperparameters.

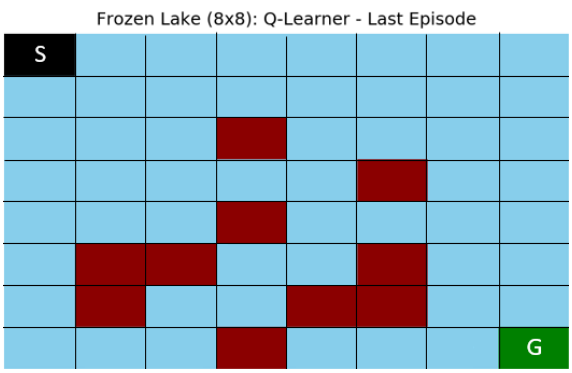


Figure 2. Frozen Lake 8x8 Environment

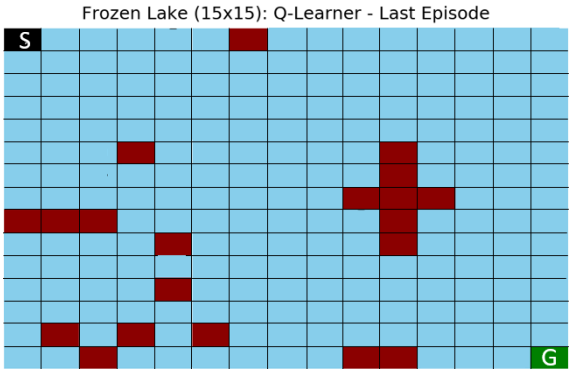


Figure 3. Frozen Lake 15x15 Environment

Markov Decision Process (MDP) Learners

Three learning algorithms will be explored when solving the MDP problems; namely, these are value iteration, policy iteration, and Q-Learning. Characteristics of each leaner are discussed below:

The value iteration (VI) algorithm calculates the utility values of each state and iteratively updates these values throughout the experiment using the Bellman equation. Utility values are influenced using a reward structure and the goal of the algorithm is to provide utility values for each state that will maximize the accumulation of rewards for each state. By weighing the influence of the new reward for entering a state against the previously calculated utility values using a discount factor, newly updated utility values will further reflect the true utility of the state. By specifying a convergence criterion, an estimated optimal policy can be inferred if the changes in the utility of all states meets a small threshold.

Similarly, the policy iteration (PI) algorithm uses the Bellman equation to evaluate how policy changes between iterations. By checking for changes in policy rather than updating values, policy iteration can converge after fewer iterations that value iteration. Policy iteration converges when the policy does not change between iterations.

Q-Learning (QL) is a reinforcement learning technique which can be implemented without knowing the reward structure of the environment or environment itself. Q-Learning calculates the expected value, which leverages information about the subsequent states and the current state, for each state. By doing so, an agent can select the action which maximizes the expected value (Q-value) for any state it is currently in. Since the learner does not know about the environment itself, it is encouraged to take random actions to explore new states over time. The number of random actions taken by the Q-Learner is governed by a random action rate (or the epsilon parameter) and it is slowly decayed through episodes so random actions will not influence the agent as the learner gains a greater understanding of the environment. Furthermore, an alpha parameter ($0 \leq \alpha < 1$) is implemented in the learner to weigh the affect of the newly learned values for entering a state against the previously calculated Q-value. A large alpha will put more emphasis on the rewards for entering the present state, while a small alpha will put more emphasis on previously calculated expected values.

Experiment Setup and Hyperparameters

The hyperparameters for the cliff walking and frozen lake environments can be seen in Table 1 below.

Table 1. Environment Hyperparameters

	Cliff Walking	**Frozen Lake**
Goal Reward	+100	+100
Step Reward	-5	-1
Terminal State Reward (cliff or hole)	-100	-100
Stochastic Probability (wind or slip)	10%	20% (10% on each side)

Hyperparameters for value and policy iteration learners can be seen in Table 2 below. The number of trials specified signify the max number of iterations in the experiment that will be done while looking for convergence. Convergence is defined with a threshold value and the number of consecutive iterations that resulted in the same policy.

Table 2. VI and PI Hyperparameters

	Value Iteration	**Policy Iteration**
Max steps/trial	500	500
Trials/experiment	500	500
Convergence threshold	0.00001	0.00001
Convergence consecutive value	5	5
Discounts Factor [min, max, step]	[0.0, 0.9, 0.1]	[0.0, 0.9, 0.1]

Hyperparameters tested for the Q-Learner are shown in Table 3 below. A grid search across all parameters were tested in each environment to find the best solution in a set of possible hyperparameters.

Table 3. Q-Learning Hyperparameters

	Q-Learning
Max steps/episode	10,000
Max episodes	10,000
Convergence theta	0.00001
Convergence consecutive theta	5
Discounts Tested [min, max, step]	[0.0, 0.9, 0.1]
Learning Rate (alpha)	[0.1, 0.5, 0.9]
Random Action Rate (Epsilon)	[0.1, 0.3, 0.5]
Epsilon decay rate	0.0001

Experiment Analysis and Results

Output policies and convergence plots are shown below for each experiment. The policies that are represented are the estimated optimal policies given the environmental and convergence criteria specified previously, as well as, the optimal hyperparameters tested each experiment.

Cliff Walking

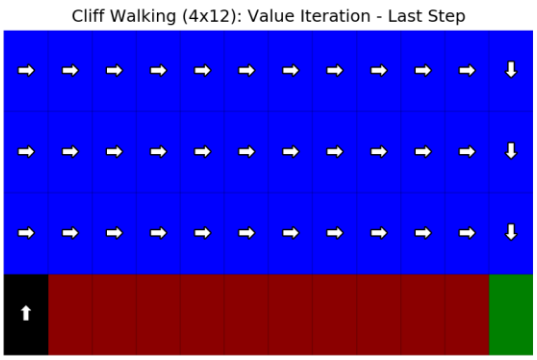


Figure 4. Cliff Walking VI Converged Policy

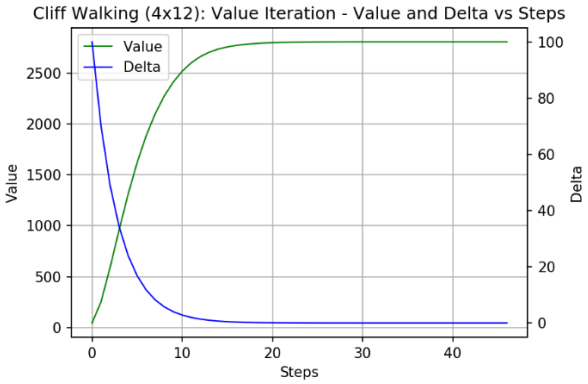


Figure 5. Cliff Walking VI Convergence Plot

Figure 4 above shows that value iteration converged with a policy whose estimated optimal action sequence is to move directly above and across the cliff. Although this policy seems risky due to the 10% chance of being blown down by wind during each step, the learner has found that taking this risk is worth taking to avoid additonal negative step rewards when moving upwards in order to reach the goal state as quickly as possible. If the step reward was lower or if the wind probability was higher (keeping all other parameters constant) the agent may have found taking the safer route across the top of the environment a better policy. Similarly, if the goal state reward was lower or the cliff state rewards were lower, it may have incentivized the agent to take the safer path across the top of the grid since the reward would have a lesser impact on the value of each state. Figure 5 above shows that this policy converged after approximately 20 iterations (labeled as steps in the figure) and a better policy was not found given the parameters of the experiment. Here, rewards remained stagnant and the change between the utility values were slowly decreasing towards zero, signifying that a better policy was not found after 50 iterations.

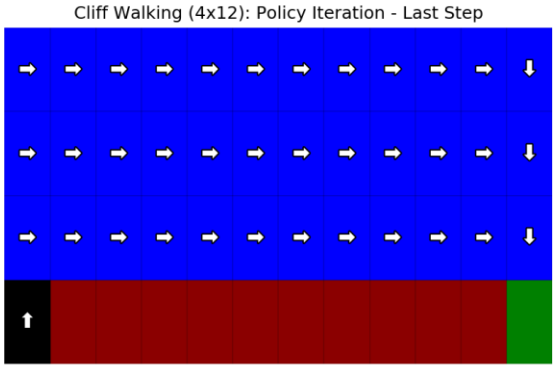


Figure 6. Cliff Walking VI Converged Policy

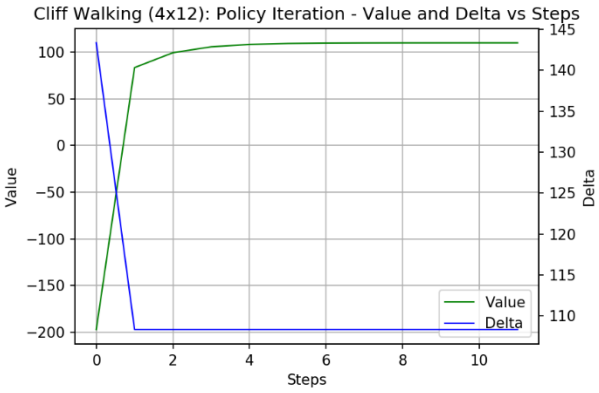


Figure 7. Cliff Walking VI Convergence Plot

From Figure 6 above, it can be seen that policy iteration converged to the same solution as value iteration. This is expected since the basis for both of these learning algorithms was derived from the Bellman equation and the goal is to improve the learner's strategy for each state until the optimal policy is discovered. Given that value iteration and policy iteration have converged to the same solution for multiple discount values, it is assumed that this is the optimal policy for this 4x12 cliff walking experiment. Figure 7 above illustrates that convergence began after 1 iteration and the learner was almost fully converged after 4 iterations. This provides further evidence that policy iteration require less iterations for convergence than value iteration since the algorithm is only looking for changes in policy rather than the convergence of utility values.

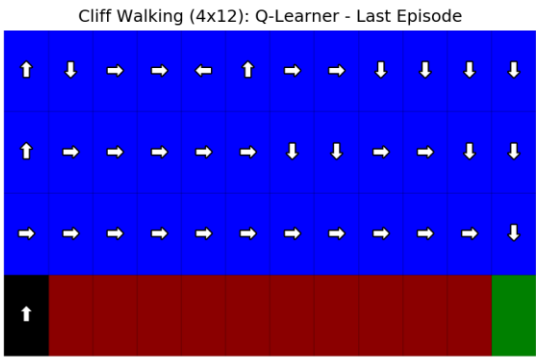


Figure 8. Cliff Walking QL Converged Policy

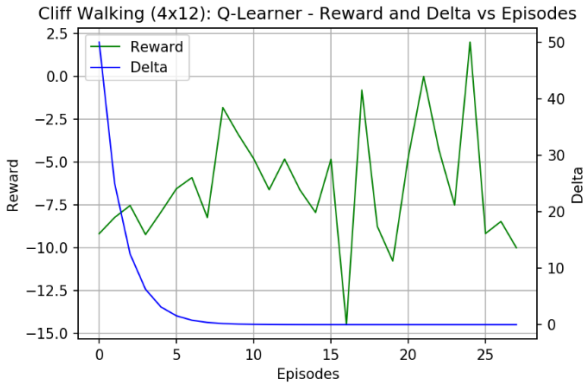


Figure 9. Cliff Walking QL Convergence Plot

Figure 8 above illustrates the output of the Q-Learner when the learning rate = 5, random action rate = 0.5, and discount rate = 0.1. The convergence criteria for this experiment specified a delta parameter of 5, meaning that if the policy has not changed after 5 episodes, the policy would be defined as converged. It can be seen that the learner has converged to a sub-optimal policy given the inefficient actions provided in the top two rows of the grid. Knowing that the policy primarily wants the agent to move directly above and across the cliff into the goal state, much like the outputs of value and policy iteration, it can be inferred that the policy for the top two rows in the environment should be the same as the policy in the third row since the environmental factors and the reward schema have remained constant. Figure 9 below shows that the learner began converging at approximately 5 episodes, but the rewards collected by the agent were sporadic due to the random actions the learner is forced to take. Given more time and stricter convergence criteria, the learner would have converged to a more optimal policy beacuase it would be forced to run across more episodes and explore more optimal state-action pairs by chance. The other hyperparameters tested with the Q-Learner resulted in less optimal results, but the general consensus in the policy was the same: the agent is instructed to move directly above and across the cliff into the goal state.

Frozen Lake 8x8

Figure 10 below shows the converged policy of the value iteration algorithm in the 15x15 Frozen Lake environment. It can be seen that the gnereral consensus of the policy is to move the agent to the right, across the board, and down to the goal state. This follows basic intuition since this strategy avoids negative

penalties (holes) and optimizes step counts towards the goal. However, it can also be inferred that the policy converged upon is sub-optimal. One example to demonstrate this is action the action suggested for the state located in row 8 (bottom row), column 3. Here, it is recommented the agent takes a step into the hole. Although this could be a viable action if the slip probability was high enough or the step reward was low enough, it is not the case in this experiment since the reward structure is shaped to move the agent towards the goal. Figure 11 illustrates that convergence began at approximately 10 iterations and almost completely converged after 25 iterations. This follows intuition that it should be longer than the cliff walking VI convergence since the problem space is larger and more random.

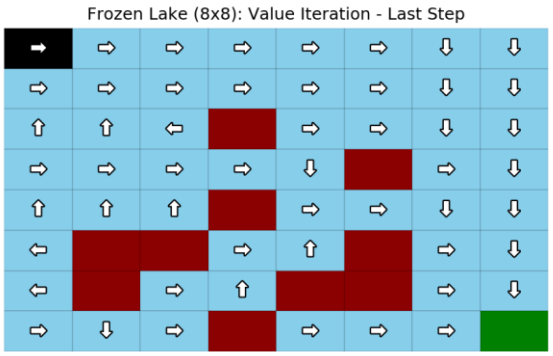


Figure 10. 8x8 Frozen Lake VI Converged Policy

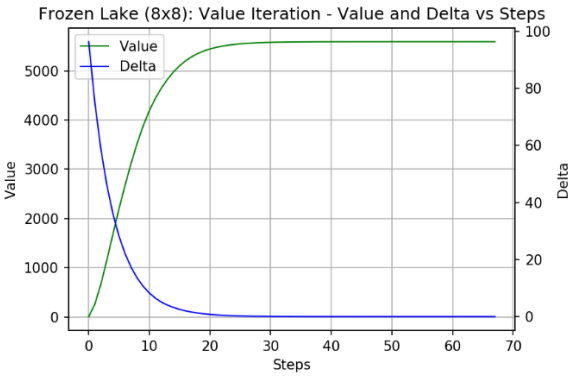


Figure 11. 8x8 Frozen Lake VI Convergence Plot

The policy that the policy iteration learner converged upon can be seen in Figure 12 below. It can be seen that the policy is similar to the output of value iteration, but slightly more optimal. Again, it is suggested that the agent moves across the board and down into the goal state, however, in areas near the holes, it is suggested that the agent moves away from the holes in a direction where the risk of slipping into a hole would be mitigated. This is particularly clear in row 2, column 4 and row 3, column 6 in the environment where it is suggested that the agent moves upwards instead of to the right, closer to the goal. Figure 13 below shows that the policy began converging after 1 iteration and the reward values converged after 5 iterations. Once again, this shows that policy iteration converged after fewer iterations than value iteration, which is to be expected since the algorithm is looking for policy convergence instead of utility value convergence.

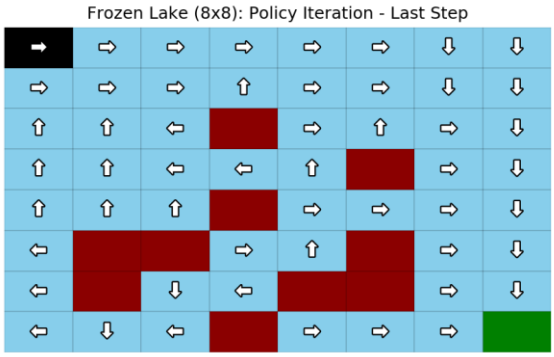


Figure 12. 8x8 Frozen Lake PI Converged Policy

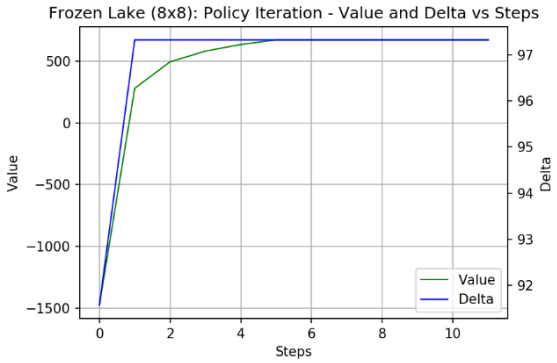


Figure 13. 8x8 Frozen Lake PI Convergence Plot

The Q-Learning output can be seen in Figure 14 below. It can be seen that the policy here follows a similar path as value and policy iteration, however the overall the policy seems to be less optimal than VI and PI learning algorithms. We can see that there are several states where it's recommended that the agent does not move despite having a slip probabilty of 20% (e.g. rows 11 and 12 in column 1). Furthermore, there are states where it is recommened that the agent ends the game by falling directly into the hole when this is detrimental to overall performance. Most notiably, this can be seen in row 8, column 5 where it would be best for the agent to move to the right and towards the goal rather than directly into the hole on it's left. Figure 15 below shows rewards and the delta between the rewards with respect to episodes. By looking at the delta and reward values, it can be seen that the learner did not converge even after 10,000 episodes. It could be that the convergence criteria was too strict or more episodes are required for convergence.

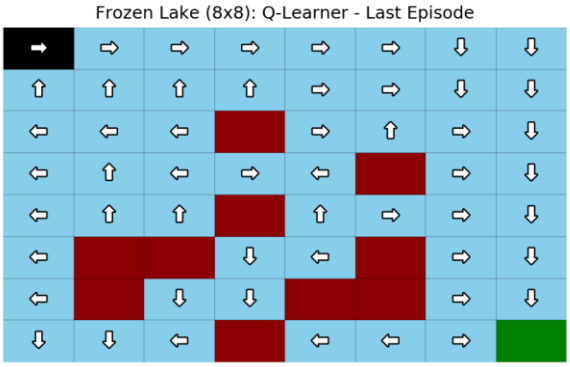


Figure 14. 8x8 Frozen Lake QL Final Policy

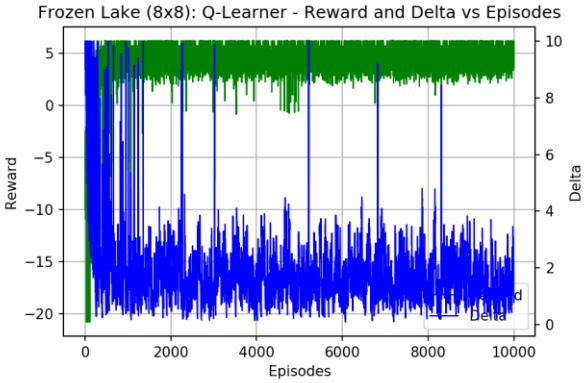


Figure 15. 8x8 Frozen Lake QL Convergence Plot

Frozen Lake 15x15

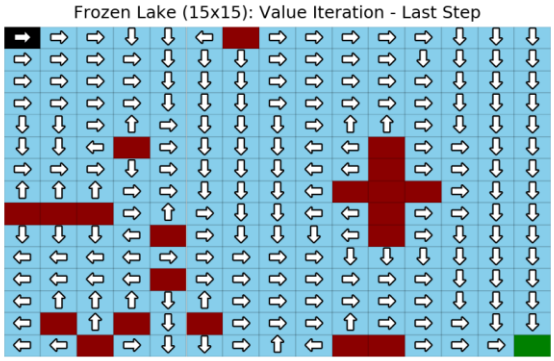


Figure 16. 15x15 Frozen Lake VI Converged Policy

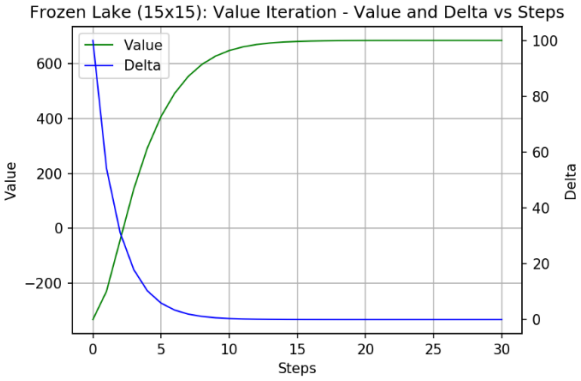


Figure 17. 15x15 Frozen Lake VI Convergence Plot

The results of value iteration on the 15x15 Frozen Lake environment can be seen in Figure 16 above. The general consensus for the algorithm is to move diagonally across the grid and into the goal state. The policy tries avoiding all hole and moving toward the goal state wherever possible. Furthermore, it accounts for the slippage probability by intentionally choosing a direction opposite of a hole to potentially slip into another state (row 15, column 6). Although this policy looks close to optimal, it can be concluded

that is not by noticing actions at particular states that are inefficient. One such state can be seen in row 15, column 5, where the agent is told to move down instead of towards the right where the probability of choosing an action that moves towards the goal state is higher than slipping in that direction. Figure 17 above illustrates that the policy began converging after 15 iterations. Surprisingly, this is lower than the number of iterations it took for the 8x8 Frozen Lake environment to converge. It is suspected that the larger number of actions were tested relative to the number of states per iteration, allowing better policy per iteration.

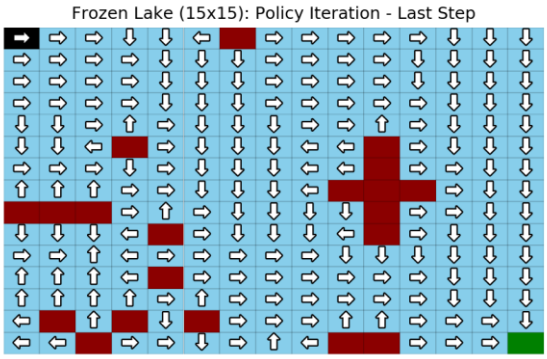


Figure 18. 15x15 Frozen Lake PI Converged Policy

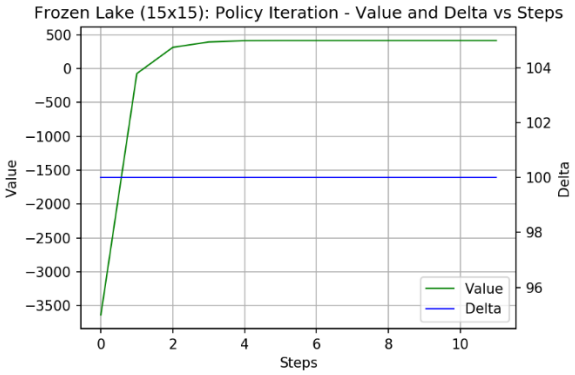
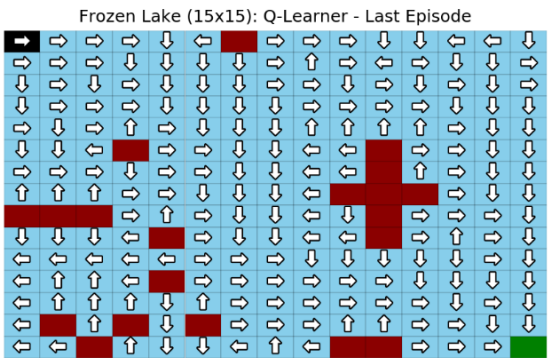


Figure 19. 15x15 Frozen Lake PI Convergence Plot

The estimated optimal policy given by policy iteration can be seen in Figure 18 above. The policy represented here is similar to the estimated VI policy in Figure 16, since the agent is instructed to move diagonally across the board into the goal state. The PI policy actions look close to optimal since they account for slip probability and mitigate risk where it can. It's difficult to conclude that PI converged exactly to the optimal policy through the visual due to the complexity of the problem, but it performed better than VI and Q-learning (discussed next) without any noticable mishaps. Figure 19 shows that PI conveged after 4 iterations which is consistent with the fact that PI will take less iterations than VI to converge. *Note: the delta in this plot should be ignored. It is a result of a bug in the code.* It should also be noted that the larger lake environment converged 1 iteration sooner than the smaller environment, this may be a attributed to the fact that there are more steps taken in the larger environment and more information for the learner to converge against.



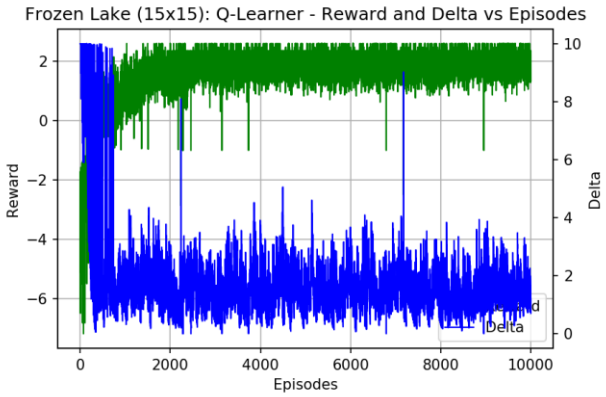


Figure 20. 15x15 Frozen Lake QL Final Policy **Figure 21. 15x15 Frozen Lake QL Convergence Plot**

Figure 20 above shows the estimated optimal policy output from the Q-Learner after 10,000 iterations. The best results from the learner were given by parameters alpha = 0.1, epsilon = 0.3, and a discount factor = 0.4. Generally, the Q-learning strategy followed the same strategy as VI and PI by moving the agent diagonally across the board from the start space. However, we can see areas where the results could be improved. Notably, the top right side of the environment directs the agent to on in varied directions when intuitively the agent should be moving downward. Furthermore, in row 15, column 4, the agent is told to move directly into a hole space which is a disadvantageous move to collect further rewards. Figure 21 above shows that the delta and reward values do not converge and is noisy throughout the learning process. In order to improve performance and reach convergence, the number of episodes can be increased so more actions can be tested through random actions.

Wall clock run times were recorded for each learner performed above. It should be noted that PI and VI had the number of trials/experiment but the number of episodes that the Q-Learner ran had to be considerablely higher in order to produce a decent policy and to attempt to reach convergence.

Table 4. Wall Clock Run-Times (s)

	Cliff Walking 4x12	Frozen Lake 8x8	Frozen Lake 15x15
Value Iteration	0.0223	0.0829	0.1030
Policy Iteration	0.3167	2.0940	7.9181
Q-Learning	0.2037	11.5480	31.4129

From Table 4 above, it can be seen that value iteration was the fasted to converge in terms of wall clock run time, followed by policy iteration, and lastly by Q-Learning. Each learner has it's advantages, as noted VI is the fastest in terms of run-time, but PI is the fastest in terms of the number of trials taken to converge. Q-Learning allows for learning without knowing the environment it is operating in, but this comes at the cost of learning time.

References

Q-learning. (2019, April 10). Retrieved April 14, 2019, from https://en.wikipedia.org/wiki/Q-learning

(2009, April 06). Retrieved April 14, 2019, from http://uhaweb.hartford.edu/compsci/ccli/projects/QLearning.pdf